EPJ Web of Conferences **245**, 07039 (2020) https://doi.org/10.1051/epjconf/202024507039
CHEP 2019

Predicting resource usage for enhanced job scheduling for opportunistic resources in HEP

Eileen Kuehn[1,*], *Max* Fischer[1], *Sven* Lange[1], *Andreas* Petzold[1], and *Andreas* Heiss[1]

[1]Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1,
76344 Eggenstein-Leopoldshafen, Germany

Abstract. To overcome the computing challenge in High Energy Physics available resources must be utilized as efficiently as possible. This targets algorithmic challenges in the workflows itself but also the scheduling of jobs to compute resources. To enable the best possible scheduling, job schedulers require accurate information about resource consumption of a job before it is even executed. It is the responsibility of the user to provide an accurate resource estimate required for jobs. However, this is quite a challenge for users as they (i) want to ensure their jobs to run correctly, (ii) must manage to deal with heterogeneous compute resources and (iii) face intransparent library dependencies and frequent updates. Users therefore tend to specify resource requests with an ample buffer. This inaccuracy results in inefficient utilisation by either blocking unused resources or exceeding reserved resources. Especially in the context of opportunistic resource provisioning the inaccuracies have an even broader impact that does not even target utilisation of resources but also composition of the most suitable resources. The contribution of this paper is an analysis of production and end-user workflows in HEP with regards to optimizing the various resources types. We further propose a method to improve user estimates.

1 Introduction

Job schedulers in high energy physics require accurate information about resource consumption of a job to find the most reasonable, available resources. For example, job schedulers evaluate information about the walltime, numbers of requested cores, or size of memory, and disk space. Jobs that use more than their requested resources are aborted or slowed down, depending on the scheduler and its configuration. Users, therefore, specify resource requests with an ample buffer to ensure that jobs are executed correctly and not canceled [1, 2]. This inaccuracy results in inefficient utilisation by either blocking unused resources, or exceeding requested and thus sometimes available resources, leading to job cancellations in the worst case. With changes to the underlying workflows, external dependencies or even the heterogeneity of resources these inefficiencies can accumulate without direct user interaction.

The scheduler can also delegate jobs to other systems by temporarily integrating opportunistic resources such as private and public clouds or HPC resources [3–5]. With the increasing demand for opportunistic resources to extend the available WLCG computing resources, the accuracy of predicted resource consumption is of particular importance. By using clouds

*e-mail: eileen.kuehn@kit.edu

such as Amazon EC2 or Telekom Cloud unnecessary costs can arise due to overestimation of resource requirements, since booked resources are billed regardless of whether they are actually used. When using resources from the research sector, which are jointly financed and used, other users cannot use resources blocked due to overestimated resource requests, although they are practically unused. Only an accurate prediction of resource consumption enables a proper selection, allocation, and integration of resources to minimise the overall costs. We, therefore, propose to improve the indicated resource consumption of end-users with predictions. This will on the one hand improve the resource utilisation in grid and cluster systems but on the other hand also the selection of proper opportunistic resources.

In this contribution, we present our results and the impact of resource predictions for both, end-user workflows and production workflows including pilot jobs. Our work focuses on resource consumption of walltime, disk, and memory but presents a generic approach that is ready for future use of other resources, such as GPUs.

2 Related Work

In [2] the authors introduce a user estimate model targeting this the issue that users tend to supply rounded values for resource estimates [1, 6]. Their findings include that only 20 different estimates are used for 90% of jobs and that the five most used walltime estimates describe 50% of the jobs considered. Based on these findings, the authors improve in [7] the walltime estimate of the user by introducing a predictor based on the mean value of the resource usage of the last two jobs of a user. They consider different window types to group similar jobs to make better predictions. They decouple the walltime prediction from the estimate by considering the prediction for job planning and the estimate as an upper limit for the actual execution of a job. The authors focus on High Performance Computing (HPC), that targets optimizing scheduling plans whereas application in HEP targets High Throughput Computing (HTC) and therefore has different requirements.

In [8] Sonmez et al. introduce four additional predictors to the mean predictor as well as various classifications of jobs. The approach is purely concerned with predicting the walltime of jobs based on measured walltime of completed jobs and, therefore, vulnerable to heterogeneous compute resources.

In [9] Gaussier et al. present a machine learning approach to improve the prediction of job walltimes. The approach establishes an asymmetric loss function, that penalizes an overestimation more than an underestimation. This approach cannot directly be adapted in the context of HTC where underestimation can lead to the re-execution of jobs and therefore increased resource consumption.

In [10] Pumma et al. introduce a job walltime prediction based on performance characteristics of short test runs of real jobs with the Linux tool perf. The measured walltime profiles are classified into different workloads by using a decision tree. To predict walltimes based on this approach means using even more resources to benchmark various jobs before execution. Furthermore pilot jobs are ambiguous and each test run might result in different walltime measurements and are therefore effectively unpredictable with this approach.

3 Use Case and Datasets

The premise of this work is that a notable portion of users overestimate the resource consumption of their jobs. This is reported in literature for other domains (cf. [1, 2]) and observable in our batch systems running HEP workflows as shown in Figure 1. An overestimation leads to a waste of resources, as they are reserved exclusively for the job, in turn reducing the throughput of a cluster or increasing the costs if commercial cloud resources are used.

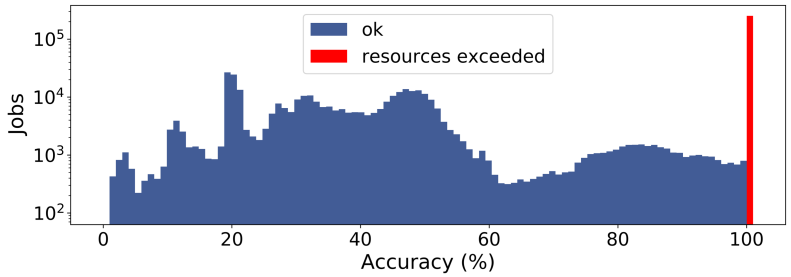

Figure 1. Accuracy of requested memory requirements to actual memory usage for end-user jobs.

An overestimation is still preferable to an underestimation, since aborting jobs that exceed their requirements likely leads to them being re-submitted by the user. This means that more than twice as many resources as required are used to execute the same job, since it is executed twice. However, if the jobs running at the same time show a massive overestimation of the resources required, these wasted resources can become more significant than re-execution of a job. It is therefore necessary to weigh up the extent to which underestimation is acceptable to avoid the double cost of aborting and re-executing the jobs.

In this paper, we consider two recorded datasets containing data about submitted jobs: the user resource estimates for these jobs, the actual resource usage of the jobs, and auxiliary metadata. The first dataset contains user jobs in the period of February 2018 to July 2018. It was recorded in a Tier 3 cluster from the Institute for Experimental Particle Physics (ETP) at KIT. The second datasets originates from the GridKa Tier 1 centre in the period from beginning of June 2018 to end of June 2018. The users usually do not directly send their jobs to GridKa but instead to a global batch system of their respective collaboration. The global batch system uses so-called pilot jobs to reserve resources in a local system, such as GridKa. The actual jobs are then executed on the resources reserved by the pilot job [11, 12]. Therefore, a direct analysis of real jobs and users is not possible for the GridKa dataset. Both datasets, the ETP and the GridKa dataset, belong to the HTC domain.

Both datasets have been cleaned up, and incomplete and incorrect entries have been removed. After the cleanup, the ETP dataset consists of 610,219 jobs and the GridKa dataset of 320,657 pilot jobs. In the following, we refer to the pilot jobs in the GridKa dataset also as jobs as we do not have sufficient information to differentiate between them.

4 Target Group-Specific Prediction of Resources

While the goal of matching requested to actual resources in similar for the HPC domain and the HTC usage in HEP, the different focus on the various kinds of resources requires modified approaches. For example, the walltime is commonly very important in HPC, as it is used to plan when future jobs can inherit a resource; in contrast, HTC is more concerned with CPUs and memory, as these define how tightly resources can be packed with jobs.

Additionally, the use case of pilot resources and opportunistic resources removes some assumptions that can be made otherwise. Pilots reduce the gain from grouping jobs by user, as a pilot represents an entire group of users. Heterogeneous resources mean that resource estimates may be based on different environments, increasing the impact of global factors to translate from one cluster to another.

The goal is to use the available resources more efficiently by estimating the resource requirements of individual jobs more accurately. A more exact estimation should lead to

Table 1. Average similarities of jobs within a cluster per clustering method and dataset. Similarity is the difference of the minimum and maximum resource usage of jobs in each workflow. Lower number means higher similarity.

	Original	User	CMD	CWD	Resources
ETP	16.3	15.3	1.7	2.0	9.9
GridKa	17.0	8.7	-	-	1.9

fewer resources remaining unused. In addition, the percentage of underestimated jobs should be reduced to prevent potential performance problems. The challenge here is that there are many, very different jobs from different users in a cluster and the resource requirements of these should be predicted.

4.1 Clustering Jobs to Workflows

To optimize resource estimation based on past jobs, it is necessary to take only comparable jobs into consideration. For jobs in HEP we can assume that jobs of the same workflow have a similar resource usage profile. It has been shown that characteristics of HEP jobs such as CPU utilisation, memory, or user can be used to cluster similar jobs and get information about the originally underlying workflows [13]. We therefore derive the clustering from such available attributes.

In total the datasets provide 23 different attributes, 13 of which are already known at the time a job is submitted, 1 at start of the job and 9 after the job has finished.[1] To ensure the clustering can be used in a production environment, only the attributes available at submission time are taken into account. These attributes include (i) the *owner* and *group* indicating who submitted the job, (ii) the *command line path* (CMD) pointing to the job executable[2], (iii) the *current working directory* (CWD) containing the path the job belongs to, and (iv) the different user's resource requests for *CPU*, *RAM*, *HDD*, or *walltime* of a job.

Based on the frequency of values of a given attribute, different approaches are feasible to cluster jobs into workflows. By default, we cluster all jobs using the fields user and group. This is based on the assumption that each user has their own recurring behavior in estimating resources and predominantly deals with the same problems. Other variants of clustering are based on CMD, CWD as well as the resources requested by the user.

Table 1 shows the quality of the clusterings compared to the initial situation per data set, without clustering. All clustered results are more similar than the original variant without clustering. To evaluate the similarity of jobs in the same cluster, the used resources of jobs are normalized and the difference between the maximum and minimum used resources summed up. The clustering for the attributes CMD and CWD are only valid for the ETP dataset as the attributes values for CMD and CWD are unique for each job within the GridKa dataset.

4.2 Optimizing User Estimates With Sliding Windows

In literature, the mean or median are often used as a predictor. However, as these predictors imply a proportion of underestimated jobs, we consider the maximum predictor instead: The maximum resource utilization of all completed jobs from the same workflow is considered

[1] Not all of the available attributes and results are considered in this publication. A comprehensive summary can be found in [14].

[2] Even though available, we exclude the arguments of the executable. In our context, these are paths of automatically generated configuration files. The relevant content is not available to use.

for the prediction. This approach is considered to ensure that few jobs are underestimated, thus providing an upper limit for the expected resource requirements.

Variations in the data can cause jobs of the same workflow to behave differently, and therefore a job may require more or less resources than another job. Care should therefore be taken to ensure that the prediction is not susceptible to individual jobs with higher resource consumption that can be considered outliers. To reduce the influence of individual outliers, not all completed jobs of a workflow are considered, but only recent jobs in a *sliding window* of size k. Outliers then only have an influence on a limited number of jobs. In order to automatically determine the best sliding window size per user in a data set (cf. [7]), we iteratively evaluate accuracy of estimates, predictions, and actual walltimes.

4.3 Local and Global User Profiles

In cases where a user starts a new workflow, there are no results available for the first jobs in this workflow with which a prediction can be inferred. This is the so-called cold start problem, which is also known from literature [15, 16]. To be able to make predictions, we introduce a local profile per user as well as a global profile for all users.

The assumption made for the local profile is that if a user underestimated or overestimated the resource requirements in previous workflows, they will also do so in new workflows. To build the local profile, the fraction of under- and overestimated jobs of the user is determined for all of his submitted jobs that have already been completed.

If the user has not yet executed any jobs, their behavior cannot be inferred. Instead, the global user profile of the cluster is considered. The global user profile works identical to the local user profile, but instead of determining the percentage of under- or overestimated jobs per user, all completed jobs of all users the cluster are analysed. The global user profile is temporarily applied until a local user profile can be established.

4.4 Evaluating the Fitness

To evaluate the different combinations of clusterings, predictors and other parameters for the two datasets, we introduce an asymmetric loss based on the amount of under- and overestimated resources.

$$\text{loss}(job, walltime) = \sum_{res} \text{weight}(job, res) \cdot |job_{res}^{\text{req}} - job_{res}^{\text{used}}| \cdot walltime$$

$$\text{weight}(job, res) = \begin{cases} 1 & job_{res}^{req} < job_{res}^{used} \\ 0.5 & \text{otherwise} \end{cases}$$

Since underestimating reduces performance by causing too many jobs to run on the same server and may lead to cancelled and rerun jobs, our loss function rates overestimation as half as significant as underestimation. The loss function is additionally weighted by the walltime of the job to consider the integrated loss over time.

To compare the influence of different variants on the same dataset we use the formula introduced in [7]:

$$\text{accuracy}(A, B) = \begin{cases} B/A & B \leq A \\ A/B & A > B \end{cases} \tag{1}$$

A stands for the requested value, be it the user's estimate or the prediction. *B* stands for the value used, that is, the resources actually used by the job. No distinction is made between overestimates and underestimates. Accuracy can be used to find out for a job, a user or the entire data set how good the predictions of the user or procedure are overall.

5 Experiment and Results

The results of different optimization schemes for resource predictions for the two datasets are shown in Tables 2 and 3. Scenario 1 combines the maximum predictor with the orginal dataset. Scenario 2 combines Scenario 1 with the four different clustering methods. Combining the clustering with the maximum predictor already gives some improvements in accuracy and number of underestimated jobs. Selection by CWD yields best results, likely because it reflects the structure imposed by users. However, every selection comes at the cost of significantly overestimating resources in case of end-user analysis. Thus, this is not suitable for making good predictions without further improvements due to the effect that individual outliers can have on the result of the optimization and consequently on the evaluation.

Scenario 3 further adds a sliding window on top and therefore focuses on minimizing the influence on outliers. The results show, that the effect of individual outliers on both data sets could be significantly reduced by using the sliding windows. The loss for all four clusterings have improved significantly for production as well as end-user jobs. At this stage, the approach is already good enough to autonomously identify similar workflows (via resources) as opposed to the structure imposed by users (via CWD).

Finally, in scenario 4 the local and global user profiles are applied. The effect of using local and global user profiles depends on the type of workflow. In the ETP dataset, the loss could not be further improved. At the cost of a higher proportion of underestimated jobs, a higher level of accuracy was achieved. In the GridKa dataset, however, significant improvements were achieved for two of the clusterings, but the overall loss of the improved clusterings is still lower than the best loss for this dataset. The use of local and global user profiles should therefore be considered on a case by case basis.

6 Conclusions and Discussion

We have shown that it is advisable to optimize the users' resource estimates, since the previous accuracy of the estimates is on average not very high. We have shown this not only for the walltime of a job, but also exemplary for memory and disk. This has a positive effect on the operator of a cluster as well as on the users in particular and high energy physics in general, as it can lead to jobs being processed faster. Specifically, to improve resource estimation, we have introduced clustering, maximum predictor, sliding window, and local and global user profiles for cold start scenarios. We have compared different combinations of these approaches and have shown that the estimates for both production and end-user jobs can be improved. Furthermore, we have shown that as the quality of resource estimation increases, fewer resources are left unused due to overestimation. Additionally, by reducing underestimated jobs, potential performance problems can be avoided.

These studies form a good basis for further improving the use of opportunistic resources based on the improved resource estimates. Although the current implementation for resource prediction already considers various types of resources, it does not consider specific weighting. This currently leads to a distortion of calculated loss as the value of memory, disk, and walltime is equalized. Future implementations should consider a weighting based on relevance and should further consider normalization of the different units of resources.

Table 2. Detailed statistics for original request and clustering in the ETP dataset when using the ideal sliding window per user, determined on 80% of each user's jobs, combined with the use of the local and global user profile to make predictions for new users or new workflows. Lower values for the rating and percentage of underestimated jobs and higher values for accuracy are better.

Scenario		Accuracy (%)			Loss	Underestimated Jobs (%)		
		RAM	HDD	Walltime	(10^{12})	RAM	HDD	Walltime
1	Original	63.4	27.8	52.5	114.9	40.9	17.2	20.2
2	User	62.6	27.9	13.3	589.0	**1.0**	**1.0**	**1.0**
	Resources	70.4	42.9	21.4	439.6	2.5	3.6	3.5
	CMD	69.9	43.9	23.7	433.3	2.2	2.1	3.7
	CWD	**74.0**	**51.4**	33.1	277.6	4.3	4.5	6.4
3	User	89.9	81.8	61.5	129.2	**8.5**	14.3	**15.3**
	Resources	**91.1**	**83.8**	**62.7**	116.0	8.7	14.8	15.8
	CMD	85.7	77.4	62.0	126.2	7.9	**13.2**	**15.3**
	CWD	85.5	77.1	62.4	**107.8**	9.1	13.7	16.1
4	User	89.8	81.7	61.4	129.8	**8.3**	14.2	**15.5**
	Resources	**91.0**	**83.8**	62.6	119.4	8.6	14.9	16.2
	CMD	90.5	83.5	62.6	126.1	11.7	**13.6**	18.0
	CWD	90.2	83.4	**63.2**	**108.5**	13.0	14.6	18.9

Table 3. Detailed statistics for original request and clustering in the GridKa dataset when using the ideal sliding window per user, determined on 80% of each user's jobs, combined with the use of the local and global user profile to make predictions for new users or new workflows Lower values for the rating and percentage of underestimated jobs and higher values for accuracy are better.

Scenario		Accuracy (%)			Loss	Underestimated Jobs (%)		
		RAM	HDD	Walltime	(10^{12})	RAM	HDD	Walltime
1	Original	51.0	92.0	-	9.5	16.0	**0.00**	-
2	User	11.5	11.9	-	430.9	**0.4**	0.3	-
	Resources	44.7	**92.7**	-	78.5	2.5	2.1	-
	CMD	**51.0**	92.0	-	**9.5**	16.0	**0.0**	-
	CWD	**51.0**	92.0	-	**9.5**	16.0	**0.0**	-
3	User	52.2	34.6	-	43.8	9.3	13.6	-
	Resources	**75.3**	**95.5**	-	**6.0**	**7.8**	12.4	-
	CMD	51.0	92.0	-	9.5	16.0	**0.0**	-
	CWD	51.0	92.0	-	9.5	16.0	**0.0**	-
4	User	52.2	34.6	-	43.8	9.3	13.6	-
	Resources	**75.0**	**95.6**	-	**6.0**	**8.8**	13.9	-
	CMD	54.0	94.2	-	7.4	54.3	62.6	-
	CWD	54.0	94.2	-	7.4	54.3	62.6	-

References

[1] A. W. Mu'alem and D. G. Feitelson, Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling, in *IEEE Transactions on Parallel and Distributed Systems* **12(6)**, pp. 529-543 (2001)

[2] D. Tsafrir, Y. Etsion, D.G. Feitelson, Modeling User Runtime Estimates, in *JSSPP 2005* **3834**, (2005)

[3] P. Marshall, K. Keahey and T. Freeman, Elastic Site: Using Clouds to Elastically Extend Site Resources, *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 43-52 (2010)

[4] M. Fischer, E. Kuehn, M. Giffels, M. J. Schnepf, A. Petzold, A. Heiss, *Lightweight dynamic integration of opportunistic resources*, EPJ Web of Conferences CHEP 2019 proceedings (to be published)

[5] M. Giffels, et al. *Effective Dynamic Integration and Utilization of Heterogenous Compute Resources*, EPJ Web of Conferences CHEP 2019 proceedings (to be published)

[6] C. B. Lee, Y. Schwartzman, J. Hardy, A. Snavely, Are User Runtime Estimates Inherently Inaccurate?, In: *JSSPP 2004*, **3277**, (2005)

[7] D. Tsafrir, Y. Etsion, and D. G. Feitelson, Backfilling Using System-Generated Predictions Rather than User Runtime Estimates, *IEEE Trans. Parallel Distrib. Syst.*, **18(6)**, 789–803 (2007)

[8] O. Sonmez, N. Yigitbasi, A. Iosup, and D. Epema, Trace-based evaluation of job runtime and queue wait time predictions in grids, in *Proceedings of the 18th ACM international symposium on High performance distributed computing (HPDC '09)*, 111–120 (2009)

[9] E. Gaussier, D. Glesser, V. Reis and D. Trystram, Improving backfilling by using machine learning to predict running times, *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1-10 (2015)

[10] S. Pumma, W. Feng, P. Phunchongharn, S. Chapeland and Tiranee Achalakul, A runtime estimation framework for ALICE, *Future Generation Computer Systems* **72**, pp. 65-77 (2017)

[11] P. Love, M. Alef, S. Dal Pra, A. Di Girolamo, A. Forti, J. Templon, E. Vanvakopoulos, JPCS, **898**, 092005 (2017)

[12] M. Turilli, M. Santcroos, and S. Jha, A Comprehensive Perspective on Pilot-Job Systems. *ACM Comput. Surv*. **51(2)**, (2018)

[13] E. Kuehn *Online analysis of dynamic streaming data*, (2018) doi:10.5445/IR/1000083227

[14] S. Lange *Analyse und Vorhersage von Nutzungsverhalten und Ressourcenverbrauch in einer wissenschaftlichen Cloud-Umgebung*, (2018)

[15] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, Learning Attribute-to-Feature Mappings for Cold-Start Recommendations, *ICDM '10*, 176–185 (2010)

[16] E. Serral, P. Valderas, and V. Pelechano, Improving the Cold-Start Problem in User Task Automation by Using Models at Runtime, in: *Pokorny J. et al. (eds) Information Systems Development*, (2011)

[17] G. Aad, J. Butterworth, J. Thion, U. Bratzler, P. Ratoff, R. Nickerson, J. Seixas, I. Grabowska-Bold, F. Meisel, S. Lokwitz et al., Jinst, **3**, S08003 (2008)

[18] J. Kennedy, C. Serfon, G. Duckeck, R. Walker, A. Olszewski, S. Nderitu et al., JPCS, **219**, 072039 (2010)